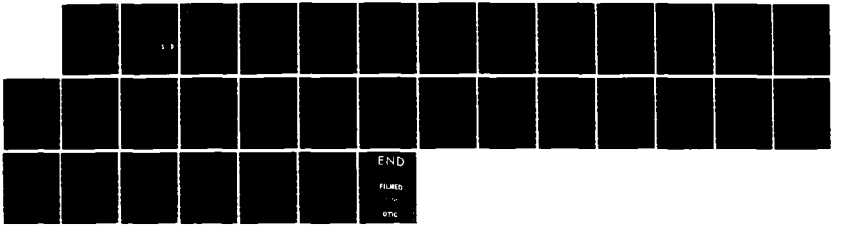
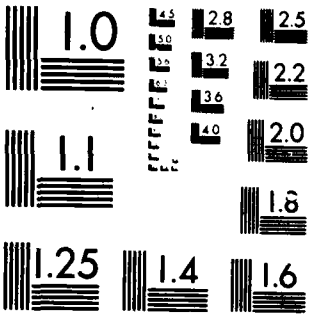
AD-A165 152 PROGRAM FLOW ANALYZER VOLUME 1(U) ARMY ELECTRONIC PROVING GROUND FORT HUACHUCA AZ E L ANDERSON SEP 85 1/1

UNCLASSIFIED F/G 9/2 NL





UNCLASSIFIED

AD-A165 152

AD No.
TECOM Project No. 7-CO-PB4-EP1-001

METHODOLOGY INVESTIGATION

FINAL REPORT

PROGRAM FLOW ANALYZER

VOLUME I

BY

EDWARD L. ANDERSON

US ARMY ELECTRONIC PROVING GROUND
Fort Huachuca, Arizona 85613-7110

SEPTEMBER 1985

DISTRIBUTION UNLIMITED

US ARMY TEST & EVALUATION COMMAND
Aberdeen Proving Ground, MD 21005-5055

UNCLASSIFIED

DEPARTMENT OF THE ARMY
HEADQUARTERS, U.S. ARMY TEST AND EVALUATION COMMAND
ABERDEEN PROVING GROUND, MARYLAND 21005-5055

REPLY TO
ATTENTION OF

AMSTE-TC-M

8 JAN 1986

SUBJECT: Methodology Investigation Final Report, Program Flow Analyzer, TECOM Project No. 7-CO-P85-EP0-001

Commander
U.S. Army Electronic Proving Ground
ATTN: STEEP-MT-DA
Fort Huachuca, AZ 85613-7110

1. Subject report is approved.

2. Test for the Best.

FOR THE COMMANDER:

GROVER H. SHELTON
C, Meth Imprv Div
Technology Directorate

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER TRMS No. 7-CO-PB4-EP1-001	2. GOVT ACCESSION NO. AD-A165 152	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Methodology Investigation Final Report Program Flow Analyzer Volumes I, II, and III		5. TYPE OF REPORT & PERIOD COVERED Methodology Investigation 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Edward L. Anderson		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS US Army Electronic Proving Ground Fort Huachuca, Arizona 85613-7110		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS US Army Test and Evaluation Command ATTN: AMSTE-AD-M Aberdeen Proving Ground, MD		12. REPORT DATE September 1985 13. NUMBER OF PAGES 201
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Software Quality
Software Metrics
Software Measurement
Automated Software Assessment

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

The Program Flow Analyzer (PFA) methodology investigation developed an automated tool for software analysis in such a manner as to simplify the effort required to develop new front-end encoders. While the PFA design has allowed a significant reduction in development time for new front-end encoders, there is a need for a more flexible multi-lingual capability. It is recommended that the Multi-lingual Static Analysis Tool be developed to satisfy this need. Keywords:

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

TABLE OF CONTENTS

VOLUME I

SECTION 1. SUMMARY

SECTION 2. DETAILS OF INVESTIGATION

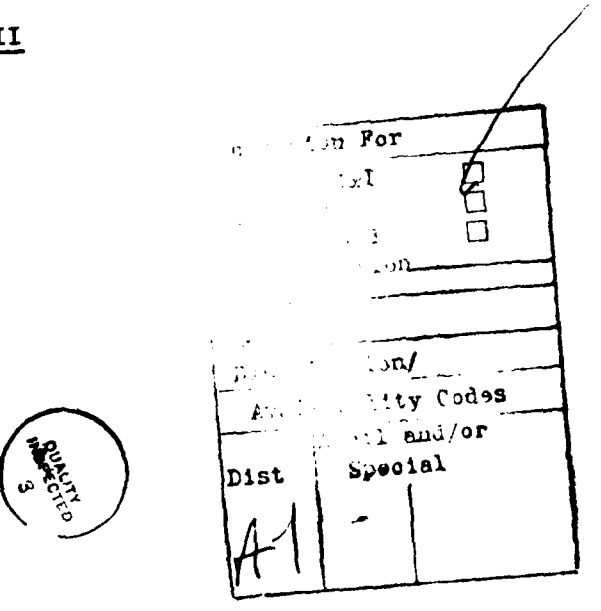
SECTION 3. APPENDIXES

VOLUME II

ACAP TECHNICAL USER'S MANUAL

VOLUME III

FCAP TECHNICAL USER'S MANUAL



SECTION 1. SUMMARY

1.1 BACKGROUND

a. The Army and DoD are continuing to develop, procure, and field automated command, control, communications, and intelligence (C^3I) systems. There is a definite need to insure that the software associated with these systems conforms to accepted standards. A manual attempt to perform the necessary tests to establish system conformance is impractical. An automated software tool to perform these tests proves to be the only viable method.

b. The Program Flow Analyzer (PFA) is a software analysis system which obtains information pertaining to software quality. The information obtained includes: McCabe's cyclomatic complexity factor, Halstead's software science metrics, number of comments, executable and nonexecutable statements, macro usage and structure diagrams. This information is then utilized by an analyst to determine the quality of the software system under test.

c. Before this information about the software is obtained, the front-end encoder of PFA must be able to correctly parse the given software language. A different front-end encoder must be used for each different language tested. Because of the many different processors and associated languages that could benefit from testing by PFA, there is a need for the PFA system to have the capability to handle many different languages. From past experience, the development of these front-end encoders can take up to 12 man-months.

1.2 OBJECTIVE. The objective of this project was to develop PFA automated tools to aid in the analysis of software source code. The design of these tools was to be in such a way as to modularize the system. This modularization would minimize the effort necessary to develop new front-end encoders. The rest of the system would be designed to have a standard format interface that could be reused with little or no modification for new languages.

1.3 SUMMARY OF PROCEDURES

a. First, the Assembly Code Analysis Program (ACAP) was developed with the capability to analyze 6800 assembly language. Other assembly languages (68000, MACRO-11, SKC) were added as needed.

b. A need developed for testing systems written in FORTRAN. The FORTRAN Code Analysis Program (FCAP) was developed to fulfill this need. The initial version of FCAP had the capability to analyze VAX FORTRAN. Later FCAP versions could analyze SKC, PDP-11/34 and RATFOR FORTRANs as well.

c. Initial applications of these software tools were used to determine whether the appropriate software metrics were being obtained and for analyzing the report writer format to determine if any further modifications were needed to assure that the metrics were presented in a clear and concise manner. Also, the data gathered was manually verified to validate the reports.

1.3.1 PFA Development

a. The first approach to reduce the development time for an encoder utilized a SRL(1) parser generator with standard reduction, analysis, and reporting functions. This method produced poor results because the limits of the tables in the parser generator were exceeded and the speed of the system was too slow.

b. The next approach used a table-driven assembly code analysis program where the op codes (mnemonics) for a specific assembly language are entered into a structured table because it has been observed that most assembly languages are similar in syntax except for differences in mnemonics. When a new language capability is needed, a new table is generated containing the new mnemonics. Minor coding changes would accompany the new mnemonic table. The initial version of the Assembly Code Analysis Program (ACAP) was finished in FY 84.

c. Along with the concept for the table-driven approach, a generic report writer was developed. The front-end encoders obtain data from the software system under test and place the data in a standard format master file. The report writer then reads this master file and generates reports based upon the information in them. With this method, any PFA tool developed at a later time that used the proper format in its master file could also use the existing report writer.

d. The FORTRAN Code Analysis Program was developed to meet the need for analysis of various dialects of FORTRAN. FCAP was not developed using the table-driven approach because FORTRAN does not lend itself easily to this method. However, a highly modular design method was used to facilitate future modifications for additional dialects of FORTRAN. The front-end encoder was designed to write the data gathered into the standard format master file so that the existing report writer could be used by the FCAP system.

e. A study was conducted as to what data and software metrics should be collected to provide information on the software under test. Many code counting metrics have been included. Also, two metrics that provide information on the structure and complexity of the software are used. These are McCabe's cyclomatic complexity factor and Halstead's software science metrics. The two were found to be quantifiable and readily automated.

1.3.2 PFA Use in Testing

a. The first version of ACAP had the capability to analyze the 6800 assembler. The 6800 capability was needed to test software for the Bradley Trainer. Next, a 68000 assembler capability was added to the ACAP system for support of testing TRAILBLAZER software. MACRO-11 assembler was added to the system for the Remotely Piloted Vehicle (RPV). In order to support testing of software for the Joint Tactical Information Distribution System (JTIDS) and RPV, an SKC assembler capability was developed.

b. The first version of FCAP was developed to analyze VAX FORTRAN. Since FORTRAN 77 and PDP-11/34 FORTRANs are essentially subsets of VAX FORTRAN, the initial version of FCAP had the capability to analyze these as well. The FORTRAN 77 capability has been used to test the Test Item Stimulator (TIS) software while the PDP-11/34 capability has been used to test RPV software. Next, a SKC FORTRAN capability was added to the FCAP system for use on testing JTIDS and RPV software. The latest version of the FCAP system had an encoder for RATFOR added to it to analyze software for TRAILBLAZER.

c. Once the PFA tools were used on test projects, the data gathered was closely examined to determine its accuracy as well as its adequacy.

1.4 SUMMARY OF RESULTS

1.4.1 PFA Development

a. Two PFA software assessment tools have been developed under this methodology investigation. First, there was ACAP, which has the capabilities listed below:

Assembly Language	FY Completed
6800	84
68000	85
MACRO-11	85
SKC	85

b. The first version of ACAP supported 6800 assembler. Because it was the initial version, it took the longest time to develop. The use of a table-driven design simplified the effort for later versions of ACAP. The two main points of design that simplified later versions were the use of a mnemonic table and the generic report writer.

c. There are two basic forms of assembler: those assembly languages that support one operand per operator and those that support two operands per operator. Once these two basic versions had been completed (6800 and 68000), any new language capabilities added took much less time to develop. The 6800 assembler (one operand per operator) was developed for the first version of ACAP. The 68000 assembler (two operators per operand) was the second version developed. The next two languages developed for the ACAP system (MACRO-11, SKC) were developed with only the generation of new mnemonic tables and minor code modifications of the 68000 assembler parsing code.

d. A generic report writer was developed for the ACAP system. The concept behind the report writer was that it was to obtain its data from a standard format master file. This master file is the output of each of the encoders in any of the code analysis systems. Therefore, as long as each new encoder developed outputs its data in the proper format, the report writer

could be used with few or no modifications. This report writer generates the reports listed in table I-1.

TABLE I-1. REPORTS GENERATED BY REPORT WRITER

Report	Comments
Module	Lists module names, the number of procedures in each module, the procedure names, and the page number of the procedure in the sequenced paginated listing.
Summary and Compliance	Indicates the degree of compliance that the software being analyzed has with selected metrics.
Software Quality Metrics	Itemizes all the metrics listed in paragraph 2-1-2 for each procedure and for the module.
Calls	Lists all the calls from each procedure within a module.
Structure Diagrams	Generated for each procedure showing up to 14 levels of procedure calls.
Undefined Externals	Lists all the unresolved external references in each procedure.
Variable Usage	Generated for both local and global variables. Lists whether a variable is modified, referenced, tested, used as an argument, or defined.
Sequenced/Paginated Listing	Generated for each procedure numbering each source line and printing a page number on each page.
Macro Usage Report	Lists all Macro definitions and usage found in the instrumented source.

e. FCAP was essentially a parallel effort to ACAP. The FCAP system is not table-driven because FORTRAN is not easily handled with this type of approach. FCAP was designed to be able to use the report writer from ACAP. This decreased the amount of coding time necessary to develop FCAP. FCAP was developed with a modular design to simplify the addition of later versions.

f. The initial version of FCAP was developed with a capability to test VAX FORTRAN. The next two capabilities of FCAP (SKC FORTRAN and RATFOR) took considerably less time to develop because of the modular design of the FCAP system. Each of the later versions took less than half the development time of the original effort. The present FCAP system has the capabilities listed below:

FORTRAN Language	FY Completed
VAN (FORTAN 77, PDP-11/34)	85
SKC	85
RATFOR	85

g. The investigation of McCabe's cyclomatic complexity factor resulted in an automated procedure for determining its value. The cyclomatic complexity factor can be used to determine the complexity of software under test with regards to control structure.

h. Software metrics obtained from applying Halstead's software science provide additional insight into the complexity of software. The metrics provided by Halstead's software science give an indication of the complexity of the software source code based upon a count of the operators and operands within the source code. The metrics include: program length, program volume, program level, potential volume, difficulty, and effort.

1.4.2 PFA Use in Testing

The PFA tools have been used to test the software of projects as listed in table I-2 below. With each application of a PFA tool, the reports listed in table I-1 were provided to the software analyst.

TABLE I-2. USE OF PFA TOOLS

Project	PFA Tool	Language(s)
BRADLEY TRAINER	ACAP	6800
TRAILBLAZER	ACAP	68000
	FCAP	RATFOR
RPV	ACAP	MACRO-11, SKC
	FCAP	SKC, PDP-11/34 FORTRANs
JTIDS	ACAP	SKC Assembler
	FCAP	SKC FORTRAN
TIS	FCAP	FORTRAN 77
ACAP/FCAP	FCAP	VAX FORTRAN

1.5 ANALYSIS

1.5.1 PFA Development

a. The development of the initial ACAP version took, by far, the longest time of all the PFA tools. This is not surprising, since many new features were being incorporated into the system and the report writer had to be developed for this initial version. The original version of the ACAP system had the capability to analyze the 6800 assembler. The next language capability to be developed for ACAP was 68000 assembler. While the 68000 capability did take less development time than did the 6800 capability of ACAP, the time savings was not as great as would later be achieved. The 6800 assembler uses one operand per operator and the parser was designed accordingly. The 68000 assembler, though, uses two operands per operator. This required the development of a parser that would allow for two operands per operator instead of one. Therefore, while it took considerably less time to develop the 68000 capability of ACAP than it did for 6800, the required development of a new parser did increase the development time. Once this parser was developed though, the next two capabilities of ACAP (MACRO-11 and SKC) achieved considerable development time savings.

b. Since the FCAP system was very different from ACAP, its development time was an increase over the time required to make additions to the ACAP system. Since FCAP was developed, the additional capabilities to the FCAP system require a great deal less time. In fact, these times closely matched the times for the additions to the ACAP system.

c. At this point, the findings indicate that the present design of the PFA tools has been successful in meeting the objective of decreasing the development time for new languages. The amount of time saved greatly depends on whether the new capability to be developed would be an addition to an existing system or would require the design of a new Code Analysis Program

(CAP) system. FCAP was designed as a new CAP system. The development of FCAP took a great deal more time than did the addition of the MACRO-11 capability to the ACAP system.

d. As a result of the investigations into McCabe's and Halstead's software assessment metrics, it has been determined that these software metrics do provide specific quantifiable properties of the software under test. McCabe's cyclomatic complexity factor has specific criteria associated with it, while Halstead's software science metrics do not. Halstead's metrics will have to be studied further in order to develop definite standards.

1.5.2 PFA Use in Testing

PFA has proven to be a useful tool to software analysts working on the test projects listed in table I-2. The reports listed in table I-1 have been used to assist the software analyst in the manner described below.

a. The Module Report provides the analyst with a table of contents of the software under test. With this report, the software analyst can find any procedure in the sequenced/paginated listing.

b. The Summary and Compliance Report allows the analyst to apply his own parameters to selected metrics.

c. The Software Quality Metrics Report provides a complete listing of all the information described in section 2.1.2 including Halstead's software science metrics. The analyst may view reports for each procedure as well as for the entire module.

d. The Calls Report provided to the software analyst assists him in structure or path analysis.

e. The structur diagrams provide software structure information to the software analyst.

f. The Undefined Externals Report assists in obtaining a list of calls to procedures whose code is not within the software under test.

g. The Variable Usage Report allows the analyst to closely check the usage of any variable within the module.

h. The Sequenced/Paginated Listing is provided to give the software analyst access to the source code, complete with page and line numbers.

i. The Macro Usage Report is used in the ACAP system to provide the software analyst with information regarding macro definitions and their usage within the source code.

1.6 CONCLUSIONS

a. Experience has shown that the design of the PFA tools has reduced development time for new front-end encoders when the new capability is an addition to an existing CAP system.

b. While the development time for new front-end encoders has been significantly reduced with the design of the present PFA tools, the development of a new CAP system still takes a significant amount of time. This indicates the need for a system that has a more flexible multi-lingual capability.

1.7 RECOMMENDATIONS

It is recommended that a study be undertaken to establish criteria for Halstead's software science metrics. In addition, it is recommended that an even more flexible tool be developed which uses as input the language description for virtually any language. This provides the advantages that the capability for any new language can be added, whether this new language is similar to previously prepared languages or not. Such a tool is the Multi-Lingual Static Analysis Tool currently being developed by USAEPG.

SECTION 2. DETAILS OF INVESTIGATION

2.1 PFA DEVELOPMENT

The effort to develop the PFA tools centered around two main efforts. The first was the actual design and development of the tools. The other was the finalizing of the decision as to what software metrics to use in the PFA tools.

2.1.1 PFA Tool Development

2.1.1.1 ACAP

a. ACAP was the first of the PFA tools developed and also took the greatest length of time. Care was taken in the design of ACAP to ensure that system would allow easy replacement or modifications to modules. This simplifies obtaining a new language capability for the system. The result of this design effort was a table-driven system. This method allows for all the mnemonics of the language to be put into a table that is referenced by the software. A new language only requires the generation of a new table along with minor code modification. Also, a report writer was desired that could be used by the PFA tools that would be required later. This required the design of a standard format master file where all the data and metrics collected would be stored by the static analyzer and later read by the report writer.

b. In all, the ACAP system is divided into four modules.

(1) The input processor converts the assembly source code to a form usable by the rest of the ACAP system and allows the user to control the system.

(2) The code translator reads each line of code, parses it into a set of tokens, and compares the tokens to information about the operators and instructions of the assembly language being used.

(3) The static analyzer calculates the software quality metrics as well as path and variable usage information.

(4) The report writer creates all the reports listed in table I-1.

c. The documentation of ACAP consists of the ACAP Technical User's Manual and commented source code. The manual is provided in Volume II. The source code, written in VAX FORTRAN, has an explanatory prologue for each subroutine as well as extensive comments throughout the executable code.

2.1.1.2 FCAP

a. The FCAP system, while not designed as a table-driven system, was designed in a manner that would allow the development of additional FORTRAN languages with much less effort than the original one. The FCAP system was designed such that all the software metrics gathered are stored in the standard format master file that can be read by the ACAP report writer. The FCAP system consists of four modules.

(1) The input processor accepts input information from the terminal on the following:

Program name
Module name
Source file name
Security classification
Master file name

(2) PASS1 performs the function of expanding include files within the source code.

(3) The Static Analyzer reads the output from PASS1 to produce information about the software being analyzed. Software quality metrics are calculated by the static analyzer as well as structure and variable usage information. The results of this processing are written to the master file which is utilized as input data to the report writer.

(4) The report writer processes the data in the master file to generate all its reports except the sequenced/paginated listing which uses the output from PASS1. The report writer in FCAP produces the same reports as in the ACAP system except for the Macro Usage Report.

b. The documentation for FCAP consists of the FCAP Technical User's Manual and commented source code. The manual is provided in Volume III. The source code, as in the ACAP source code, has an explanatory prologue for each subroutine as well as extensive comments throughout the executable code.

2.1.2 Software Metrics

a. On the basis of an earlier investigation of software metrics and early runs of the PFA tools, a decision was made to use McCabe's cyclomatic complexity factor and Halstead's software science metrics.

b. McCabe's cyclomatic complexity has numeric qualities associated with it. The cyclomatic number is obtained by searching the software source code for branches, loops, and logical statements. The cyclomatic complexity number can be used to determine whether the source code under test has excessive branching or looping structures. Higher cyclomatic numbers are a good indication that the procedure under test needs to be broken down into multiple procedures with a less complex structure. A simpler structure leads to increased understandability and maintainability. Software with McCabe's complexity numbers of greater than 10 are considered to have excessive branching and looping.

c. Halstead's software science metrics also produce numeric properties of the software. Halstead's software science metrics differ from McCabe's cyclomatic complexity in that there are no established criteria to apply to these metrics, once obtained. The metrics provided by Halstead's are based on the count of operators and operands in the procedure under test. These operator and operand counts are applied to several different formulas that yield metrics related to the quality of the software source code under test.

d. The software quality metrics report in the CAP system itemizes all the metrics found in the following listing. A further explanation of these metrics can be found in appendix C. Halstead's metrics are calculated from the number of operators and operands.

1. Number of lines in the abstract.
2. Number of comment lines.
3. Number of executable lines.
4. Number of non-executable lines.
5. Number of lines.
6. Number of comments for executable lines.
7. Number of comments for non-executable lines.
8. Number of in line comments.
9. Percent of executable lines commented.
10. Percent of non-executable lines commented.
11. Maximum consecutive lines w/o comment.
12. Number of executable statements.
13. Number of non-executable statements.
14. Number of multi-line statements.
15. Number of multi-statement lines.
16. Number of entry points.
17. Number of exit points.
18. Number of forward branches.
19. Number of backward branches.
20. Number of branches out of routine.
21. Number of conditional branches.
22. Number of unconditional branches.
23. Number of nodes.
24. Number of paths.

25. McCabe's cyclomatic.

26. Number of lines skipped.

27. Number of includes.

28. Number of unique operators.

29. Number of operators used.

30. Number of unique operands.

31. Number of operands used.

32. Lines of code (primary language).

33. Lines of code (embedded language).

34. Primary versus embedded language.

35. Number of variables in a procedure.

2.2 PFA USE IN TESTING

a. ACAP

(1) The ACAP system has been used as a software analysis tool to aid software analysts on several projects. The first capability developed for ACAP, 6800 assembler, was used to assess software for the Bradley Trainer.

(2) Part of the TRAILBLAZER system uses a 68000 processor. The 68000 capability of the ACAP system was developed specifically to be used on TRAILBLAZER 68000 software. ACAP has now been used on multiple versions of software of TRAILBLAZER 68000 assembler. All the report capabilities of the report writer were used to aid analysts in their testing of TRAILBLAZER 68000 software.

(3) The SKC Assembler capability for the ACAP system was developed to support JTIDS. That software is currently being tested by ACAP. Upon completion of testing, the results will be given to software analysts for inspection.

(4) The RPV project uses assembler software in the form of MACRO-11 assembler. The MACRO-11 capability of the ACAP system was developed to support this project. Each version of RPV MACRO-11 assembler is tested by ACAP to provide feedback as to the quality and degree of compliance to standards of the software. The RPV also has SKC assembler associated with it; ACAP has been used to test this software.

b. FCAP

(1) The initial version of the FCAP system had the capability to test VAX FORTRAN. This capability was used to test the source code of both the ACAP and FCAP systems. This was done to determine how the code for the two PFA tools compared to standards.

(2) Software for TIS is written in FORTRAN 77. Since FORTRAN 77 is essentially a subset of VAX FORTRAN, no modification to FCAP was required to test TIS software. FCAP has been used to test the version of TIS software that was used at the start of the TIS Format Qualification Testing (FQT). It is planned to be used on the software at the end of FQT also.

(3) RPV uses two different FORTRANS (SKC and NORDEN 11/34). The software for the NORDEN 11/34 is the same as the PDP 11/34. This version of FORTRAN was able to be tested by the FCAP system with no modifications to the system. FCAP has now been used to test two different versions of RPV software written in PDP-11/34 FORTRAN. RPV also uses SKC FORTRAN. An SKC FORTRAN capability was added to the FCAP system to support this need. FCAP has been used to test the latest version of SKC FORTRAN for RPV.

(4) The TRAILBLAZER system also uses RATFOR. This required the development of an almost entirely new encoder. The RATFOR capability has been developed and used on RATFOR software from the TRAILBLAZER system.

(THIS PAGE INTENTIONALLY BLANK)

SECTION 3. APPENDICES

(THIS PAGE INTENTIONALLY BLANK)

APPENDIX A. METHODOLOGY INVESTIGATION PROPOSAL

DEPARTMENT OF THE ARMY Mr. Deaver/brt/AUTOVON
HEADQUARTERS, U.S. ARMY TEST AND EVALUATION COMMAND 283-3677
ABERDEEN PROVING GROUND, MARYLAND 21005

REPLY TO ATTENTION OF
DRSTE-AD-M

SUBJECT: MMT Methodology Improvement Program Directive, Program Flow Analyzer, TRMS No. 7-CO-PB4-EP1-001

Commander
US Army Electronic Proving Ground
ATTN: STEEP-MT-T
Fort Huachuca, AZ 85613

1. Reference:

a. TECOM Regulation 70-12, dated 1 June 1973.

b. AR 700-90, dated 15 March 1982,

c. DARCOM Handbook 11-4.1-82, Manufacturing Methods and Technology Exhibit P-16 Handbook, dated December 1981.

d. Manufacturing Methods and Technology Instructional Guide for Preparing Project Status Reports (DRCMT 301), dated November 1982.

2. This letter and attached STE Forms 1188 and 1189 (Enclosure 1) constitute a directive for the subject investigation under the TECOM Methodology Improvement Program BP 5397-5071.

3. The MMTP at Enclosure 2 is the basis for headquarters approval of the subject investigation.

4. Special Instructions:

a. All reporting will be in consonance with paragraph 9 of reference 1a. The final report, when applicable, will be submitted to this headquarters, ATTN: DRSTE-AD-M, in consonance with Test Event STE Form 1189.

b. Semiannual Project Status Reports RCS-DRCMT-301, Manufacturing Technology (MANTECH) Program, are to be provided to this headquarters by 15 February and 15 August for each year that the investigation is active. The information contained in the RCS-301 Report is entered into a data bank by the Industrial Base Activity (IBEA), Rock Island, Illinois, and used by DARCOM to monitor the progress of the program. Therefore, the information must be provided

DRSTE-AD-M
SUBJECT: MMT Methodology Improvement Program Directive, Program Flow Analyzer, TRMS No. 7-CO-PB4-EP1-001

in the exact format shown in reference 1d. If the investigation is supported with funds for more than one fiscal year, it must be reported for each year.

c. Recommendations of the new TOPs or revisions to existing TOPs will be included as part of the recommendation section of the final report. Final decision on the scope of the TOP effort will be made by this headquarters as part of the report approval process.

d. The addressee will determine whether any classified information is involved and will assure that proper security measures are taken when appropriate. All OPSEC guidance will be strictly followed during this investigation.

e. Prior to test execution, the test activity will verify that no safety or potential health hazards to humans participating in testing exist. If safety or health hazards do exist, the test activity will provide a safety/health hazards assessment statement to this office prior to test initiation.

f. Environmental documentation for support tests or special studies is the responsibility of the test activity and will be accomplished prior to initiation of the investigation/study.

g. Upon receipt of this directive, test milestone schedules as established in TRMS II data base will be reviewed in light of known other workload and projected available resources. If rescheduling is necessary, a letter citing particulars, together with recommendations, will be forwarded to Commander, US Army Test and Evaluation Command, ATTN: DRSTE-AD-M, with an information copy to DRSTE-TO-O, within 15 calendar days.

h. The HQ, TECOM point of contact is Mr. Larry W. Miller, ATTN: DRSTE-AD-M, AUTOVON 283-2170/2375.

i. FY84 MMT funds in the amount of $75,000 have been authorized for this investigation, DARCOM Form 1006 will be forwarded by the Comptroller.

FOR THE COMMANDER:

2 Encl
as

GROVER H. SHELTON
C, Meth Imprv Div
Analysis Directorate

EXHIBIT P-16 (PART I)
Production Engineering Measures (PEM) Project
RCS DRCMT-835

Date: 1983

SUBTASK No. 113

---

1. Project Number: 0855071 2. Fiscal Code: PA. 5397 3. Cost: $100K

4. Project Title: Program Flow Analyzer (PFA) Tools for Production of Computer/Software System Specific Encoders

5. Name and location of facility/contractor: US Army Electronic Proving Ground, Fort Huachuca, AZ 85613

6. General Objective: Improvement of manufacturing processes, techniques and equipment.

7. Problem: The Army and DoD are continuing to develop, procure and field automated command control, communications, and intelligence (C^3I) systems. These automated systems utilize different processors and software languages. Automated tools are needed to reduce the time and manpower required for the production of system specific encoders to interface the production systems software to the tools which are evaluating the product software.

8. Proposed Solution: This task will improve the necessary methods for automating the generation of system specific encoders. The task will look at, in a top down approach, the design requirements for system specific encoders. Automated tools will be specified, designed, tested, and documented to reduce the time and effort required to generate encoders to evaluate software in production systems which are automated.

9. Justification: The very nature of software allows it to be easily changed to either correct problems, or to add capabilities based on new requirements or experience. Changes to a software configuration must be controlled, well documented and validated. The most efficient, practical way to do this is through the use of automated tools. Experience has shown that encoder generation can consume anywhere from 3 to 12 months, depending on the complexity of the system under test. The goal of this investigation is to reduce the encoder development time by a factor of two or three.

10. Benefits:

a. Quantifiable benefits (S/I): None Basis:

b. Non-quantifiable benefits: This task will reduce the manhours required to generate an encoder by a factor of two or three; thus improving production testing capability.

11. Deliverables:

a. Reports identifying encoder needs for automated tools.

b. Computer programs in both hard copy and machine readable form.

c. Updated maintenance and user's manuals.

d. Report of methods and tools validation.

encl 3

12. Funding Profile and Scheduled Technical Completion Dates:

Fiscal Year	$Costs, (XK)	Month-Year
FY 85	100	Sep 85
FY 86	100	Sep 86
FY 87	100	Sep 87
TOTAL		

13. End Items Supported:

a. Primary - Effort will support the production testing of software on all automated syste
b. Secondary - The quality of the software directly affects the performance, maintainabili and supportability of all automated systems.

14. Key Milestone Dates:

a. PEP Completion for primary end item - N/A
b. MMT Completion - September 1986
c. Primary End Item TC - N/A
d. Start of Full Scale Production - N/A
e. Preliminary Design Criteria for Facility - N/A

15. Related MMT and Feasibility Demonstration Efforts:

a. Project Nos.	None			
b. Initiation Date				
c. Completion Date				

16. Plan for Implementation of Efforts' Results:

a. When - FY85/86 (limited capability); FY87 (complete capability)
b. Where - USAEPG
c. How - Automated tools for creating system specific encoders will be used to decrease manpower and time requirements.
d. Who - TECOM Field Activities responsible for testing and evaluating software.
e. Cost - No addition costs are anticipated.

- Energy Resource Impact Statement: This study does not involve resources beyond those required for study, analysis, and computer program generation; therefore, no impact is expected on energy resources.
- Project Engineer:

a. Name - Larry W. Miller
b. Organization - Cdr, US Army Test & Evaluation Command, ATTN: DRSTE-AD-M, APG, MD 21005
c. Phone Numbers, AUTOVON 283-2375 Commercial 301-278-2375

Enclosure 1 - Environmental Documentation

None required

DEPARTMENT OF THE ARMY Mr. Deaver/brt/AUTOVON
HEADQUARTERS U S ARMY TEST AND EVALUATION COMMAND 283=3677
ABERDEEN PROVING GROUND, MARYLAND 21005

REPLY TO ATTENTION OF
DRSTE-AD-M

8 FEB 1984

SUBJECT: Change to Test Directives

Commander
US Army Electronic Proving Ground
ATTN: STEEP-MT-T
Fort Huachuca, AZ 85613

1. Reference letter, DRSTE-AD-M, 31 Jan 84, subject: MMT Methodology Improvement Program Directive, Program Flow Analyzer, TRMS No. 7-CO-PB4-EP1-001.

2. Paragraph 2 of the reference directive should be deleted and replaced with the following:

"2. This letter constitutes a directive for the subject task under the TECOM Methodology Improvement Program funded by 650808E8635."

FOR THE COMMANDER:

for GROVER H. SHELTON
C, Meth Imprv Div
Analysis Directorate

APPENDIX B. ACRONYMS

ACAP	Assembly Code Analysis Program
ADP	Automatic Data Processing
CAP	Code Analysis Program
FCAP	FORTRAN Code Analysis Program
HOL	High-Order Language
JTIDS	Joint Tactical Information Distribution System
PFA	Program Flow Analyzer
RPV	Remotely Piloted Vehicle
TIS	Test Item Stimulator

(THIS PAGE INTENTIONALLY BLANK)

APPENDIX C. DESCRIPTION OF SOFTWARE QUALITY METRICS

1. Number of Lines in the Abstract. The number of all comment lines between either the start of a file or a preceding end statement and the next non-commented line.

2. Number of Comment Lines. The number of all comment lines, excluding those in the abstract, found in the file.

3. Number of Executable Lines.

4. Number of Non-Executable Lines.

5. Number of Lines - The sum of executable lines, non-executable lines, and non-prologue comment lines.

6. Number of Comments in Executable Lines. The number of comments found in the executable section of the code. The executable section consists of all source code from the first executable line to the end of the procedure.

7. Number of Comments in Non-Executable Lines. The number of comments in the non-executable section of the code. The non-executable section consist of the source code found before the first executable line in a procedure.

8. Number of In-Line Comments. The number of comments found on the same line as some code.

9. Percentage of Executable Lines Commented. A percentage based on the number of comments located in the executable section of code and the number of executable statements.

10. Percentage of Non-Executable Lines Commented. A percentage based on the number of comments located in the non-executable section of code and the number of non-executable statements.

11. Maximum Consecutive Lines Without Comments. The maximum number of consecutive lines in the executable section that are not commented.

12. Number of Executable Statements. The number of executable statements in the code.

13. Number of Non-Executable Statements. The number of non-executable statements in the code.

14. Number of Multi-Line Statements.

15. Number of Multi-Statement Lines.

16. Number of Entry Points. The number of entry points in a given subroutine.

17. Number of Exit Points. The number of exit points in a given subroutine.

18. Number of Forward Branches.

DESCRIPTION OF SOFTWARE QUALITY METRICS (CONT)

19. Number of Backward Branches. The number of goto's to a previous area of code.

20. Number of Branches out of Routine.

21. Number of Conditional Branches. The number of goto's in a subroutine associated with an IF statement.

22. Number of Unconditional Branches. The number of forward goto's in a subroutine that are not associated with an IF statement.

23. Number of Nodes. The number of nodes to be used to calculate McCabe's cyclomatic complexity.

24. Numnber of Paths. The number of paths to be used to calculate McCabe's cyclomatic complexity.

25. McCabe's Cyclomatic. McCabe's cyclomatic complexity, calculated by the formula: paths - nodes + 2. This assumes single and exit points.

26. Number of Lines Skipped.

27. Number of Includes. The number of include statements encountered.

28. Number of Unique Operators. The number of unique operators found in the subroutine.

29. Number of Operators Used. The total number of operators used in the subroutine.

30. Number of Unique Operands. The number of unique operands found in the subroutine.

31. Number of Operands Used. The total number of operands used in the subroutine.

32. Lines of Code (Primary Language). The number of lines of primary code found in the subroutine. A system tested by FCAP has FORTRAN as the primary language. Embedded within the FORTRAN is a data base query language. This query language is considered embedded code.

33. Lines of Code (Embedded Language). The number of lines of embedded code found in the subroutine.

34. Primary versus Embedded Language Switches. The number of switches between primary code and embedded code.

35. Number of variables in a procedure - The total number of variables used in a subroutine.

APPENDIX D. DISTRIBUTION LIST

Addressee	Number of Copies
Commander US Army Test and Evaluation Command	
ATTN: AMSTE-TC-M	3
AMSTE-TO	2
AMSTE-EV-S	1
AMSTE-TE	6
Aberdeen Proving Ground, MD 21005-5055	
Commander Defense Technical Information Center ATTN: DTIC-DDR Cameron Station Alexandria, VA 22314-5000	2
Commander US Army Aberdeen Proving Ground ATTN: STEAP-MT-M Aberdeen Proving Ground, MD 21005-5000	2
Commander US Army Yuma Proving Ground ATTN: STEYP-MSA Yuma, AZ 85364-5000	2
Commander US Army Jefferson Proving Ground ATTN: STEJP-TD-E Madison, IN 47250-5000	1
Commander US Army Dugway Proving Ground ATTN: STEDP-PO-P Dugway, UT 84022-5000	1
Commander US Army Cold Regions Test Center ATTN: STECR-TM APO Seattle 98733-5000	1
Commander US Army Electronic Proving Ground ATTN: STEEP-TM-AC Fort Huachuca, AZ 85613-7110	4

Addressee	Number of Copies
Commander US Army Tropic Test Center ATTN: STETC-TD-AB APO Miami 34004-5000	1
Commander US Army White Sands Missile Range ATTN: STEWS-TE-PY STEWS-TE-O STEWS-TE-M STEWS-TE-A White Sands Missile Range, NM 88002-5000	4 1 1 1

END

FILMED

4-86

DTIC